<u>In the Claims</u>:

Please amend claims 10, 24, and 26, and cancel claims 40 and 51, all as shown below.

1 - 9. (Canceled)

10. (Currently Amended): A computer-readable medium storing instructions for processing an invocation at a dynamically generated wrapper, comprising the steps of:

receiving, from an application, an invocation by a wrapper object, the wrapper object instantiated from a wrapper class, the wrapper class extended from a superclass which implements a wrapper interface that includes a pre-invocation handler and a post-invocation handler, the invocation directed to a wrapped resource adapter;

initiating pre-processing by calling a pre-invocation handler configured to execute server-side code wherein the server-side code includes transaction processing code;

calling the wrapped object;

receiving a result from the wrapped object;

initiating post-processing by calling a post-invocation handler configured to execute post processing server-side tasks wherein the post-processing server-side tasks include transaction management; and

providing the result to the application, thereby enabling the application to access vendor specific extension methods of the wrapped resource adapter:

wherein the superclass has a member variable to hold a vendor object, a non-argument constructor to instantiate the wrapper object, and an init method to initialize the wrapper object.

- 2 -

11 – 23. (Canceled)

24.   (Currently Amended):   A computer-readable medium storing instructions for processing an invocation at a dynamically generated wrapper, comprising the steps of:

receiving, from an application, a method invocation to a resource adapter;

calling a wrapper object for processing the method invocation wherein the wrapper object is dynamically generated from a resource adapter class;

initiating pre-processing by the wrapper object, wherein the wrapper object calls a pre-invocation handler configured to perform server side logic, wherein the server-side logic includes transaction processing logic;

forwarding the method invocation to the resource adapter by the wrapper object on behalf of the application;

receiving a result of the method invocation from the resource adapter by the wrapper object;

initiating post-processing by the wrapper object, wherein the wrapper object calls a post-invocation handler configured to perform server-side logic, wherein the server-side logic ~~include~~ includes transaction management; and

providing the result to the application, thereby enabling the application to access vendor specific extension methods of the resource adapter;

wherein the wrapper object is dynamically generated at runtime by a wrapper factory on an application server.

25.   (Canceled)

26.   (Currently Amended):   A computer-readable medium storing instructions for dynamically generating a wrapper object, comprising the steps of:

receiving a resource adapter class at an application server;

performing reflection on the resource adapter class to identify interfaces implemented by the resource adapter class;

dynamically generating a wrapper class at runtime that extends from a superclass, wherein the superclass implements a wrapper interface that includes a pre-invocation handler and a post-invocation handler, and the wrapper class implements the interfaces identified through reflection;

instantiating a wrapper object from the wrapper class;

initiating pre-processing by the wrapper object, wherein the pre-processing code includes calling a pre-invocation handler, wherein the pre-invocation handler is configured to execute server-side code, wherein the server-side code includes transaction processing code; and

providing the wrapper object to an application that requires support for the interfaces implemented by the resource adapter class;

wherein the superclass has a member variable to hold a vendor object, a non-argument constructor to instantiate the wrapper object, and an init method to initialize the wrapper object.

27-28. (Canceled)

- 4 -

29.   (Previously Presented): The computer-readable medium of claim 26 further comprising:

initiating post-processing by the wrapper object, wherein post-processing including calling a post-invocation handler, wherein the post-invocation handler is configured to perform post-processing server side tasks, wherein the post-processing server-side tasks include transaction management.

30.   (Previously Presented): The computer-readable medium of claim 10, wherein the wrapper object is a proxy generated at runtime and acts as a delegate for an underlying vendor object.

31.   (Previously Presented): The computer-readable medium of claim 10, wherein the wrapper object is used to intercept method invocations from an application program to a vendor object and provide for execution of server side tasks in a pre-invocation handler and a post-invocation handler.

32.   (Previously Presented): The computer-readable medium of claim 10, wherein the wrapper object is used to intercept a method invocation against the vendor object.

33.   (Previously Presented): The computer-readable medium of claim 10, wherein the wrapper object provides for server side tasks to be performed before sending a wrapped result to the application.

34.   (Previously Presented): The computer-readable medium of claim 10, wherein the wrapper object is dynamically generated at runtime by a wrapper factory on an application server.

35.   (Previously Presented): The computer-readable medium of claim 10, wherein retrieved meta information from performing reflection allows an application server to dynamically generate a wrapper class that matches the vendor class.

36.   (Previously Presented): The computer-readable medium of claim 10, wherein a wrapper class includes all public interfaces implemented by a vendor class and required by the application.

37.   (Previously Presented): The computer-readable medium of claim 10, wherein the application can cast the wrapper object to a vendor interface to access vendor extension methods.

38.   (Previously Presented): The computer-readable medium of claim 10, wherein the application server has code for dynamically generating the wrapper.

39.   (Previously Presented): The computer-readable medium of claim 10, wherein a wrapper factory uses a static method to dynamically generate a wrapper.

40.   (Cancelled)

41.   (Previously Presented): The computer-readable medium of claim 26, wherein the wrapper object is a proxy generated at runtime and acts as a delegate for an underlying vendor object.

42.   (Previously Presented): The computer-readable medium of claim 26, wherein the wrapper object is used to intercept method invocations from an application to a vendor object and provide for execution of server side tasks in a pre-invocation handler and a post-invocation handler.

43.   (Previously Presented): The computer-readable medium of claim 26, wherein the wrapper object is used to intercept a method invocation against a vendor object.

44.   (Previously Presented): The computer-readable medium of claim 26, wherein the wrapper object provides for server side tasks to be performed before sending a wrapped result to the application.

45.   (Previously Presented): The computer-readable medium of claim 26, wherein the wrapper object is dynamically generated at runtime by a wrapper factory on the application server.

46.   (Previously Presented): The computer-readable medium of claim 26, wherein retrieved meta information from performing reflection allows the application server to dynamically generate a wrapper class that perfectly matches a vendor class.

47.   (Previously Presented): The computer-readable medium of claim 26, wherein the wrapper class includes all public interfaces implemented by a vendor class and required by the application.

48.   (Previously Presented): The computer-readable medium of claim 26, wherein the application can cast the wrapper object to the vendor interface to access vendor extension methods.

49.   (Previously Presented): The computer-readable medium of claim 26, wherein the application server has code for dynamically generating the wrapper.

50.   (Previously Presented): The computer-readable medium of claim 26, wherein a wrapper factory uses a static method to dynamically generate a wrapper.

51.   (Cancelled)